

MI FORMACIÓN DUAL EN PROPERLY SOFTWARE



CYNTIA GARCÍA RUIZ
1º DESARROLLO DE APLICACIONES
MULTIPLATAFORMA
I.E.S. CAMPANILLAS



ÍNDICE

- INTRODUCCIÓN
- OBJETIVOS
- PROYECTOS:
 - * SERVICIOS RESTFUL CON JERSEY
 - * CLIENTE RESTFUL
 - * TEST JUNIT
 - * WEB
- ENTORNOS DE DESARROLLO:
 - * ECLIPSE
 - * POSTMAN
 - * POSTGRESQL
- LENGUAJES:
 - * JSON, XML, HTML, CSS
- CONCLUSIONES

Mi empresa elegida para mi formación fue Properly Software, la razón de mi elección fue debido a que era una empresa que tenía bastante recorrido en el Parque Tecnológico. Me parecía una buena opción puesto que me podían aportar muy buenos conocimientos para mi formación, en un entorno más cercano.


Programación:

- Seguir practicando los recursos aprendidos
- Utilizar plataformas para trabajar con repositorios
- Trabajar el acceso a ficheros
- Realizar aplicaciones que accedan a una base de datos

Base de datos:

- Modelado de base de datos
- Utilizar LDM
- Practicar procedimientos y triggers

Entornos de desarrollo:

- Utilizar una o varias IDEs en el trabajo
 - Participación en reuniones de planificación
- 

Lenguaje de marcas:

- Utilizar HTML 5, CSS 3, JavaScript
- Utilizar documentos xml
- Manejo de otros lenguajes como Json...

Desde que entré en marzo hasta mitad de abril estuve aprendiendo como funcionaba un Servicio RESTful y creando el mío propio.

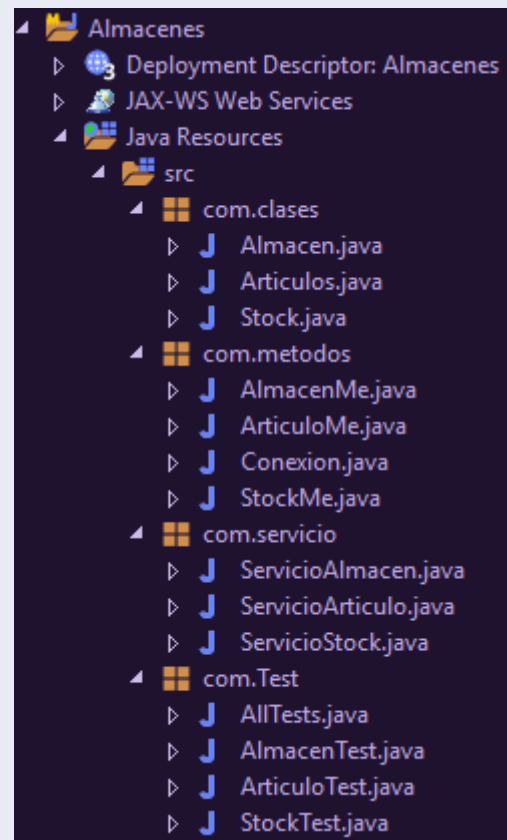
¿Que es un servicio RESTful?

Un servicio RESTful es un programa basado en la arquitectura rest.

Tiene estas características:

- Permite listar, crear, leer, actualizar y borrar información.
- Para las operaciones anteriores necesita una URL y un método HTTP para acceder a ellas.
- Usualmente regresa la información en formato JSON.
- Retorna códigos de respuesta HTML. Por ejemplo 200 (ok), 202(accepted), 201(created) o 404(not found).

El tema elegido del proyecto es la gestion de almacenes, tenía que utilizar una librería de java llamada jersey. Se utiliza para simplificar más el servicio RESTful y el cliente. El proyecto lo empece con maven, en el entorno de eclipse, este se divide en cuatro partes: clases , métodos, servicios y test.



Cree las clases correspondientes con sus getter y setter.

```
50 import org.json.JSONException;
51
52 public class Almacen {
53     private int idAlmacen;
54     private String nombreAlmacen;
55
56     public Almacen() {
57     }
58
59     public Almacen(int idAlmacen, String nombreAlmacen) {
60         this.idAlmacen = idAlmacen;
61         this.nombreAlmacen = nombreAlmacen;
62     }
63
64     public int getIdAlmacen() {
65         return idAlmacen;
66     }
67
68     public void setIdAlmacen(int idAlmacen) {
69         this.idAlmacen = idAlmacen;
70     }
71
72     public String getNombreAlmacen() {
73         return nombreAlmacen;
74     }
75
76     public void setNombreAlmacen(String nombreAlmacen) {
77         this.nombreAlmacen = nombreAlmacen;
78     }
79
80     @Override
81     public String toString() {
82         try {
83             return new JSONObject().put("id", idAlmacen).put("NombreAlmacen", nombreAlmacen).toString();
84         } catch (JSONException e) {
85             return null;
86         }
87     }
88 }
```


Después continué con los métodos, hice uno por cada funcion que quería utilizar en el servicio, get - post - put - delete - getespecifico. Utilicé jdbc para crear la conexión y así poder hacer las consultas a la base de datos y traer la información.

```
4
5 import java.sql.Connection;
11 |
12 public class AlmacenMe {
13
14     public static List<Almacen> getAlmacenes() {
15
16         try {
17             Class.forName("org.postgresql.Driver");
18             Connection conexion = Conexion.crearConexion();
19             Statement sentencia = conexion.createStatement();
20             ResultSet resultado = sentencia.executeQuery("SELECT * FROM almacenes");
21             ArrayList<Almacen> alm = new ArrayList<Almacen>();
22
23             while (resultado.next()) {
24                 alm.add(new Almacen(resultado.getInt("idalmacen"), resultado.getString("nombrealmacen")));
25             }
26             resultado.close();
27             sentencia.close();
28             conexion.close();
29             return alm;
30         } catch (Exception e) {
31             e.printStackTrace();
32         }
33         return null;
34     }
35 }
```

Una vez terminado los métodos, cree el servicio que es el que utiliza esas funcionalidades.

```
18 @Path("/almacen")
19 public class ServicioAlmacen {
20
21     @GET
22     @Produces({ MediaType.APPLICATION_JSON })
23     public List<Almacen> getAlm() {
24         return AlmacenMe.getAlmacenes();
25     }
26
27     @GET
28     @Path("/{idalmacen}")
29     @Produces({ MediaType.APPLICATION_JSON })
30     public Almacen getConcreto(@PathParam("idalmacen") int idalmacen) {
31         return AlmacenMe.getAlmacen(idalmacen);
32     }
33
34     @POST
35     @Produces({ MediaType.APPLICATION_JSON })
36     public Response addAlmacen(Almacen alm)
37         throws JSONException {
38         AlmacenMe.addAlmacen(alm);
39         return Response.status(201).build();
40     }
41
42     @PUT
43     @Produces({ MediaType.APPLICATION_JSON })
44     public Response updateAlmacen(Almacen alm) throws JSONException {
45         AlmacenMe.updateAlmacen(alm);
46         return Response.status(202).build();
47     }
48
49     @DELETE
50     @Path("/{idalmacen}")
51     @Produces({ MediaType.APPLICATION_JSON })
52     public Response deleteAlmacen(@PathParam("idalmacen") int idalmacen) {
53         AlmacenMe.deleteAlmacen(idalmacen);
54         return Response.status(200).build();
55     }
56 }
```

Desde mitad de abril hasta el 3 de mayo continué con el cliente que consume los servicios RESTful. Mientras hacía el servicio, las pruebas las hacía en Postman, que es una api para pruebas REST. El cliente lo hice con jersey y con .net.

```
package com.almacen;

import java.util.Scanner;

public class Delete {

    public static void main(String[] args) {
        String input = new Scanner(System.in).nextLine();
        Client client = Client.create();
        WebResource webResource = client.resource("http://localhost:8080/Almacenes/rest/almacen/" + input);
        String s = webResource.accept("application/json").delete(String.class);
        System.out.println(s);
        ClientResponse response = webResource.accept("application/json").delete(ClientResponse.class);
        int status = response.getStatus();
        System.out.print(status);
    }
}
```

```
public static void main(String[] args) {
    try {

        URL url = new URL("http://localhost:8080/Almacenes/rest/articulo");
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setDoOutput(true);
        conn.setRequestMethod("PUT");
        conn.setRequestProperty("Content-Type", "application/json");
        String input = new Scanner(System.in).nextLine();

        OutputStream os = conn.getOutputStream();
        os.write(input.getBytes());
        os.flush();

        if (conn.getResponseCode() != HttpURLConnection.HTTP_ACCEPTED) {
            throw new RuntimeException("HTTP code : " + conn.getResponseCode());
        }
    }
}
```

Más tarde cuando ya tenía acabado el servicio completo y habíamos comunicado las dos máquinas con los clientes de cada ordenador, empecé un **test junit** para comprobar que la api no tiene fallos y que su funcionalidad va con normalidad. El test lo que hace es comprobar los estados que devuelve nuestro servidor, cuando le hacemos una petición. Es necesario este tipo de test para asegurar la calidad de los software que creamos.

The logo for JUnit, featuring a large green 'J' and a large red 'U' followed by the word 'nit' in a smaller red font.

Después implementé el almacén a una página web con los estilos bootstrap, mediante un proyecto web dinámico en jsp, se constituye de una página inicial de login y después un crud básico de altas, bajas, modificaciones y listado. Todavía está en proceso, quería encriptar las contraseñas en el login para que no se vean los usuarios y contraseñas, cuando sacamos los usuarios de la base de datos.

Artículo

Código artículo:

Nombre:

Precio coste:

Precio venta:

Guardar

Administración de almacenes

Usuario:

Contraseña:

Entrar

Vector de Fondo creado por Skelchepedia

Project Explorer Administración almacenes.jsp Lista Artículos

http://localhost:8080/Almacen-web/articulos.jsp

Artículos

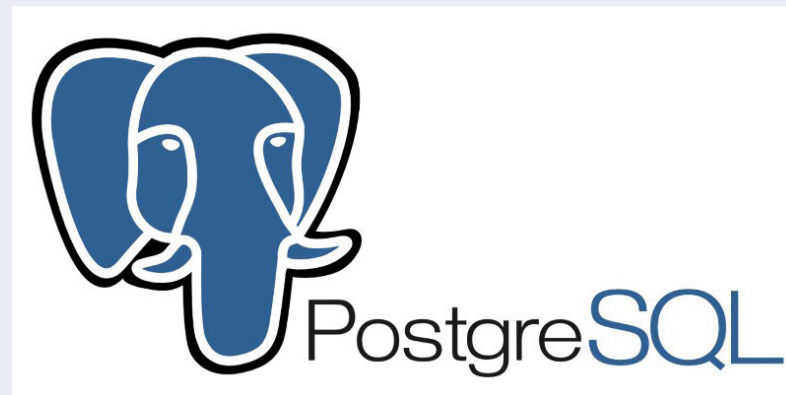
Escriba el nombre del artículo **Buscar**

Nuevo artículo

No se encontraron registros para la búsqueda

Markers Properties Servers Data Source Explorer Schemas Problems Console

Los entornos de desarrollo que he utilizado han sido eclipse, para crear los proyectos maven que utilizaba para el código java, es más automático o ligero ya que solo tenias que definir las dependencias y no tenias que descargar todas las librerías; postman que lo utilizaba para hacer peticiones al servidor de una manera mas ágil y sencilla; y postgresSQL para la base de datos.



Los lenguajes que he utilizado han sido xml, para las dependencias maven tuve que mirar como funcionaba la estructura puesto que al principio solo era copiar y pegar los archivos de los ejemplos pero, al implementar mi servicio tenia que aprender como crear el mio propio para saber añadir y quitar dependencias sin fastidiar el código; Json lo utilizaba para la información que devolvía del servidor y cuando tenía que mandar un almacen mediante post tenia que ser en un objeto Json; html junto con java ,css y bootstrap para crear la página web de los almacenes.



La formación dual ha sido una forma diferente de adquirir los conocimientos necesarios para mi aprendizaje, para mi al principio fue un choque en cuanto a que ya no tienes a alguien que te explique la lección y tengas un libro donde te viene todo bien explicado paso a paso, pero después me pareció algo genial porque vas navegando y de una cosa te vas a otra y vas cogiendo información como una esponja para después llevar a cabo tus metas. Me ha parecido interesante y enriquecedora la experiencia. He aprendido bastante sobre java, un lenguaje que es extremadamente diverso y extenso.



Muchas gracias por la atención!!!